STABILITY SELECTION
DOCUMENT WITH SOLUTIONS

Computational Epidemiology

MSc Health Data Analytics and Machine Learning

4^{th} of March 2021

CONTENTS

OVERVIEW OF THE SESSION

DATA DESCRIPTION

You have been provided with inflammatory proteomics measurements on participants from the Italian component of the European Prospective Investigation into Cancer and Nutrition (EPIC) and Norwegian Women and Cancer Study (NOWAC) cohorts. In participants from the NOWAC study, transcriptomics measurements are also available. Please refer to **S Dagnino, B Bodinier et al. Prospective identification of elevated circulating CDCP1 in patients years before onset of lung cancer, Cancer Research (2021)** for more information on the study subjects and OMICs measurements.

MATERIAL PROVIDED

You have been provided with one script "penalisation_functions.R", and two folders: "Data" and "Literature".

In "Literature", you will find the paper mentioned above.

In "Data", you will find four files:

- "Covariates_selected_proteins.rds": individual characteristics
- "Proteins_selected_denoised_re.rds": proteomics measurements after denoising on technical confounder (plate) and centre
- "Transcripts_log_transformed.rds": gene expression measurements of 142 smoking-related transcripts, as identified in a large meta-analysis (T Huan et al. A whole-blood transcriptome meta-analysis identifies gene expression signatures of cigarette smoking, Hum Mol Genet (2016))
- "Transcripts_annotation.rds": annotation of the transcripts.

Both the proteomics and transcriptomics datasets have been prepared beforehand and can directly be used for network analyses. The data preparation included: the extraction of observations/features that did not pass quality control or had more than 30% missing values, data transformation ($\log_2$-transformation for proteins, log-transformation for transcripts), imputation of missing values, denoising for technical confounders (proteins only).

OUTLINE OF THE SESSION

In this practical, we will use stability selection models to explore the associations between inflammatory proteins and smoking-related transcripts. The session can be decomposed into three parts:

1. Using the stability selection LASSO, you will identify a set of smoking-related proteins.
2. Using the stability selection graphical LASSO, you will explore the conditional independence structure between inflammatory proteins.

3. Using the multi-block extension of the stability selection graphical LASSO, you will integrate inflammatory proteins and smoking-related transcripts in a multi-OMICs network. The set of smoking-related proteins identified in step 1 will be highlighted in the network.

STABILITY-ENHANCED VARIABLE SELECTION

LOADING REQUIRED PACKAGES AND FUNCTIONS

In the following sections, we will a series of packages, as well as functions defined in a separate script:

```
LoadPackages=function(packages){
  for (i in 1:length(packages)){
    suppressPackageStartupMessages(library(packages[i], character.only=TRUE))
  }
}

LoadPackages(c("pheatmap","corpcor","abind","parallel",
               "RColorBrewer","igraph","ppcor","mvtnorm",
               "pROC","glasso","stabs","huge","pulsar",
               "QUIC","glassoFast","colorspace","glmnet"))

source("penalisation_functions.R")
```

LOADING THE DATA

The proteomics and transcriptomics data can be loaded using the code below:

```
covars=readRDS("Data/Covariates_selected_proteins.rds")
proteins=readRDS("Data/Proteins_selected_denoised_re.rds")
transcripts=readRDS("Data/Transcripts_log_transformed.rds")

ids=intersect(rownames(proteins), rownames(transcripts))
covars=covars[ids,]
transcripts=transcripts[ids,]
proteins=proteins[ids,]
print(all(rownames(proteins)==rownames(transcripts)))
```

```
## [1] TRUE
```

```
print(all(rownames(proteins)==rownames(covars)))
```

```
## [1] TRUE
```

LASSO regression

We will use packyears as a measure of smoking, and adjust the models on age and BMI (no need to account for gender as all participants are Women). These two confounders will be included in the LASSO model without penalising them. This can be done using the argument `penalty.factor` and ensures that the confounders are always selected in the LASSO models.

Challenge 1: Run the LASSO regression described above (i.e. with all proteins as predictors, packyears as outcome and adjusted on age and BMI). List the variables selected in the model calibrated by cross validation minimising the Mean Squared Error of Prediction (MSEP).

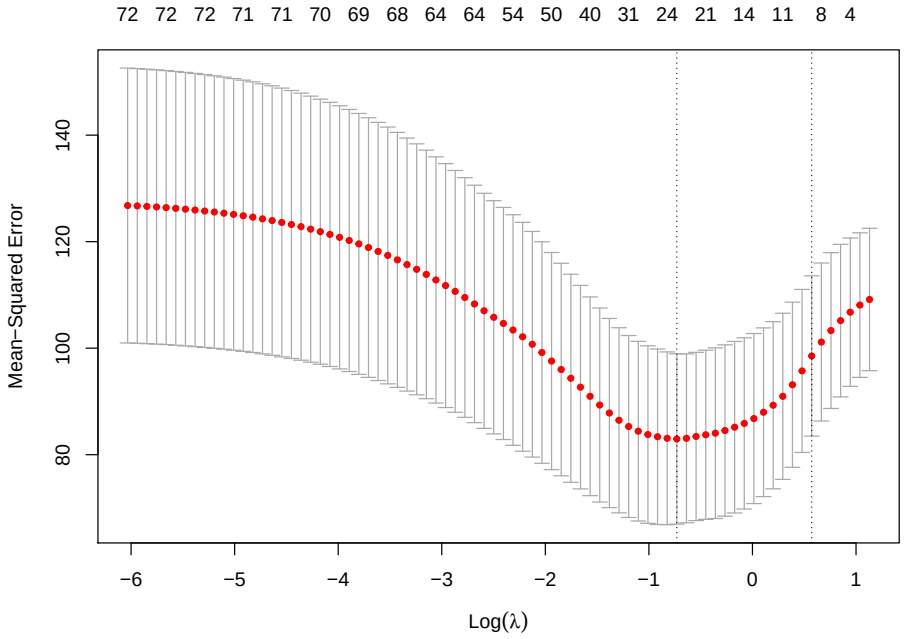
The LASSO model can be run and calibrated using the `cv.glmnet()` function from the `glmnet` package:

```
# Complete cases
ids_to_exclude=unique(c(which(is.na(covars$packyears)),
                        which(is.na(covars$age.sample)),
                        which(is.na(covars$bmi))))
if (length(ids_to_exclude)>0){
  covars=covars[-ids_to_exclude,]
}
proteins=proteins[rownames(covars),]
y=covars$packyears
x=cbind(proteins, age=covars$age.sample, bmi=covars$bmi)

# Cross-validation
set.seed(1)
t0=Sys.time()
mymodel=cv.glmnet(x=x, y=y, penalty.factor=c(rep(1,ncol(proteins)),0,0), family="gaussian")
t1=Sys.time()
print(t1-t0)
```

```
## Time difference of 0.160594 secs
```

```
plot(mymodel)
```



The set of selected proteins can be extracted as follows:

```
# Selected variables
beta_lasso=coef(mymodel, s="lambda.min")[2:(ncol(proteins)+1),]
selected_lasso=names(beta_lasso)[which(beta_lasso!=0)]
print(paste0(length(selected_lasso), " proteins are selected"))
```

```
## [1] "22 proteins are selected"
```

```
print(selected_lasso)
```

```
##  [1] "CD8A"      "CDCP1"     "OPG"       "IL17A"     "CXCL11"    "CXCL9"
##  [7] "CCL4"      "CD6"       "SCF"       "CCL11"     "FGF23"     "LIFR"
## [13] "IL18R1"    "IL12B"     "IL10"      "Flt3L"     "CXCL6"     "CCL28"
## [19] "IFNgamma" "MCP2"      "NT3"       "TWEAK"
```

If you run this model multiple times using different seeds, you might get slightly different results. In stability selection, the variable selection algorithm (the LASSO in this example) is applied on different subsamples of the data and all results are aggregated. The output of a stability selection model is less likely to vary when it is run multiple times with different seeds. It is increasingly stable with the number of iterations.

Stability selection LASSO

The stability selection LASSO can be run using the `CalibrateRegression()` function, defined in the penalisation_functions.R script. This function is internally using the `glmnet()` function to get the LASSO results for different values of the penalty parameter λ and different subsamples of the data. The `penalty.factor`

and `family` arguments can be set as in the `glmnet` package. The parameter `K` defines the number of iterations (i.e. the number of subsamples on the data on which the model is fitted), and `tau` defines the size of the subsample (0.5 is optimal according to simulation studies):

```
# Complete cases
ids_to_exclude=unique(c(which(is.na(covars$packyears)),
                        which(is.na(covars$age.sample)),
                        which(is.na(covars$bmi))))
if (length(ids_to_exclude)>0){
  covars=covars[-ids_to_exclude,]
}
proteins=proteins[rownames(covars),]
y=covars$packyears
x=cbind(proteins, age=covars$age.sample, bmi=covars$bmi)

# Running stability selection
t0=Sys.time()
out=CalibrateRegression(xdata=x, ydata=y, K=100, tau=0.5, verbose=FALSE,
                        penalty.factor=c(rep(1,ncol(proteins)),0,0), family="gaussian")
```

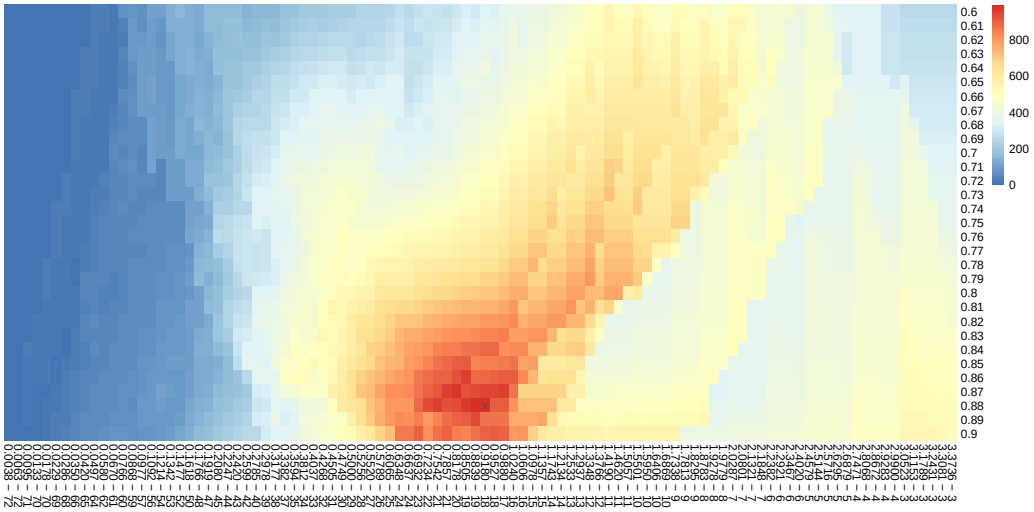
```
##   |                                                                      |
```

```
t1=Sys.time()
print(t1-t0)
```

```
## Time difference of 2.169062 secs
```

The pair of parameters (penalty λ and threshold in selection proportion π) can be jointly calibrated to maximise the stability score:

```
CalibrationPlot(out)
```



In this calibration plot, the stability score is colour-coded, the different values of the penalty λ are on the X-axis (the corresponding average number of variables selected is also reported) and different thresholds in selection proportion π are on the Y-axis. In this example, the calibrated parameters are 0.9180 (penalty parameter) and 0.88 (selection proportion threshold). The stability selection model includes the variables selected in more than 88% of the models fitted with a penalty parameter of 0.9180:

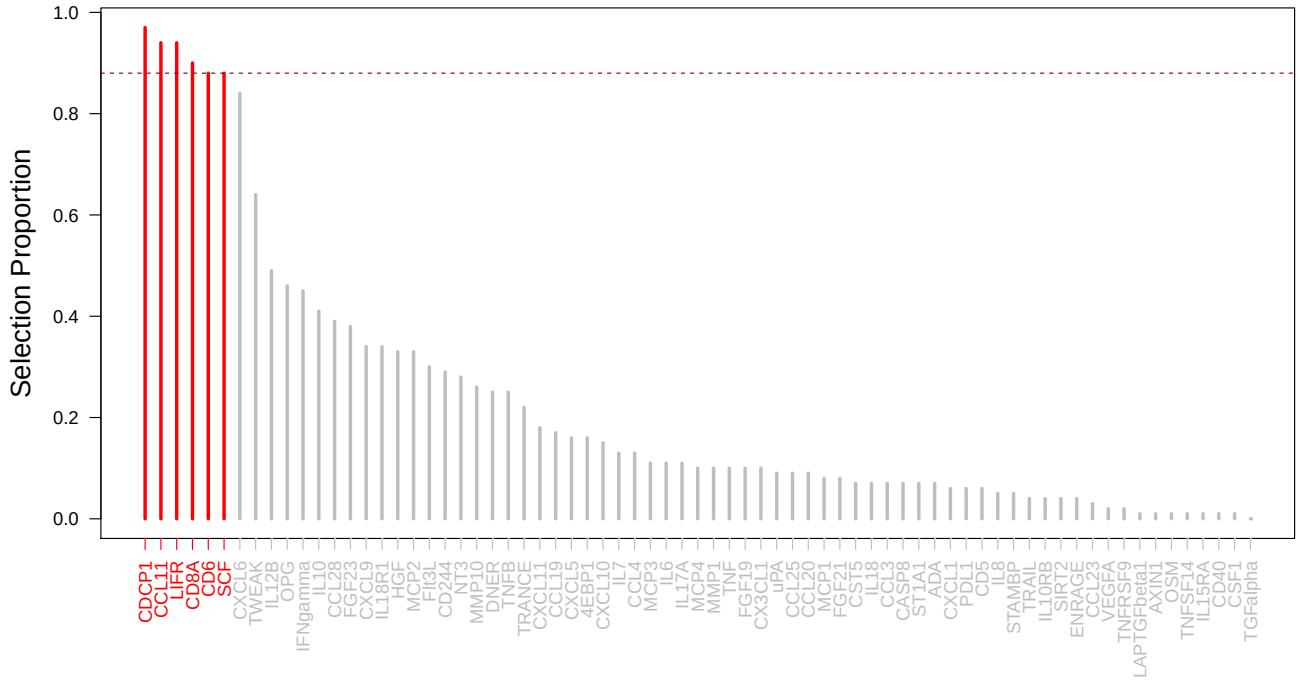
```
# Calibrated selection proportions
selprop=CalibratedSelectionProportionsRegression(out, plot=FALSE)
selprop=selprop[-which(names(selprop)%in%c("age","bmi"))]
print(selprop)
```

```
##       CDCP1       CCL11        LIFR        CD8A         CD6         SCF
##        0.97        0.94        0.94        0.90        0.88        0.88
##       CXCL6       TWEAK       IL12B         OPG    IFNgamma        IL10
##        0.84        0.64        0.49        0.46        0.45        0.41
##       CCL28       FGF23       CXCL9      IL18R1         HGF        MCP2
##        0.39        0.38        0.34        0.34        0.33        0.33
##       Flt3L       CD244         NT3       MMP10        DNER        TNFB
##        0.30        0.29        0.28        0.26        0.25        0.25
##      TRANCE      CXCL11       CCL19       CXCL5       4EBP1      CXCL10
##        0.22        0.18        0.17        0.16        0.16        0.15
##         IL7        CCL4        MCP3         IL6       IL17A        MCP4
##        0.13        0.13        0.11        0.11        0.11        0.10
##        MMP1         TNF       FGF19      CX3CL1         uPA       CCL25
##        0.10        0.10        0.10        0.10        0.09        0.09
##       CCL20        MCP1       FGF21        CST5        IL18        CCL3
##        0.09        0.08        0.08        0.07        0.07        0.07
##       CASP8       ST1A1         ADA       CXCL1        PDL1         CD5
##        0.07        0.07        0.07        0.06        0.06        0.06
##         IL8      STAMBP       TRAIL      IL10RB       SIRT2      ENRAGE
##        0.05        0.05        0.04        0.04        0.04        0.04
##       CCL23       VEGFA     TNFRSF9 LAPTGFbeta1       AXIN1         OSM
##        0.03        0.02        0.02        0.01        0.01        0.01
##     TNFSF14      IL15RA        CD40        CSF1    TGFalpha
##        0.01        0.01        0.01        0.01        0.00
```

```
# Calibrated parameters
hat_params=GetArgmax(out)
print(hat_params)
```

```
##         lambda   pi
## [1,] 0.9179824 0.88
```

```r
# Visualisation of selection proportions
par(mar=c(10,5,1,1))
plot(selprop, type="h", lwd=3, las=1, xlab="", ylab="Selection Proportion", xaxt="n",
     col=ifelse(selprop>=hat_params[2], yes="red", no="grey"), cex.lab=1.5)
abline(h=hat_params[2], lty=2, col="darkred")
for (i in 1:length(selprop)){
  axis(side=1, at=i, labels=names(selprop)[i], las=2,
       col=ifelse(selprop[i]>=hat_params[2], yes="red", no="grey"),
       col.axis=ifelse(selprop[i]>=hat_params[2], yes="red", no="grey"))
}
```



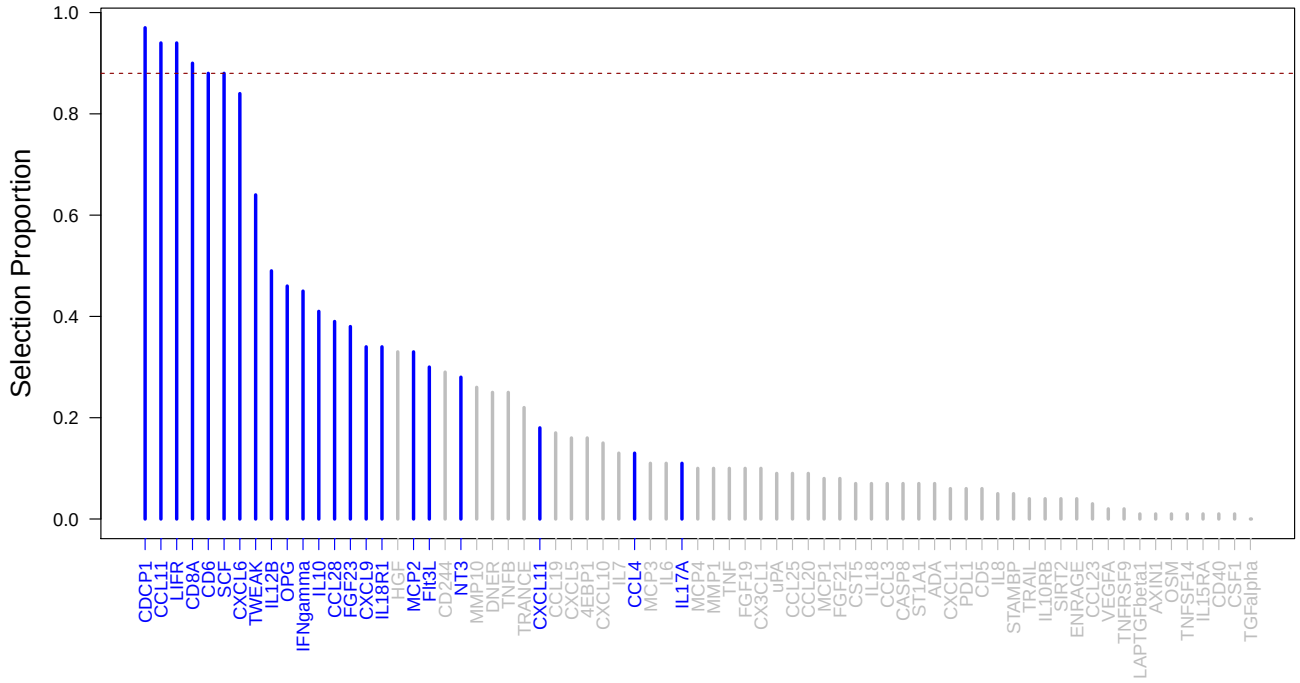
The stability selection model identified a set of six proteins (CDCP1, CCL11, LIFR, CD8A, CD6 and SCF) associated with packyears (selection proportions above 0.88).

Challenge 2: The set of proteins identified by stability selection is sparser than the set of LASSO-selected proteins. Compare the results from these two models.

The proteins that were selected in the LASSO models are highlighted in the figure below:

```r
# Visualisation of selection proportions
par(mar=c(10,5,1,1))
plot(selprop, type="h", lwd=3, las=1, xlab="", ylab="Selection Proportion", xaxt="n",
     col=ifelse(names(selprop)%in%selected_lasso, yes="blue", no="grey"), cex.lab=1.5)
abline(h=hat_params[2], lty=2, col="darkred")
for (i in 1:length(selprop)){
```

```
  axis(side=1, at=i, labels=names(selprop)[i], las=2,
       col=ifelse(names(selprop)[i]%in%selected_lasso, yes="blue", no="grey"),
       col.axis=ifelse(names(selprop)[i]%in%selected_lasso, yes="blue", no="grey"))
}
```



The proteins identified by the LASSO are generally among the ones with higher selection proportions. However, some of the signals are detected in less than 50% of the models, which indicates that these associations could be driven by outlying observations.

STABILITY-ENHANCED GRAPHICAL MODELS

In this section, we will explore the conditional independence structure among the inflammatory proteins.

UNCONSTRAINED CALIBRATION

The stability selection graphical LASSO can be applied to the proteins data using the function `CalibrateNetwork()`, defined in the script penalisation_functions.R. This model is similar to the stability selection LASSO used in the previous section. Internally, the graphical LASSO is fitted on subsamples of the data with different values of the penalty parameter λ (instead of the LASSO in the previous section). The arguments `K` and `tau` are defined as above.

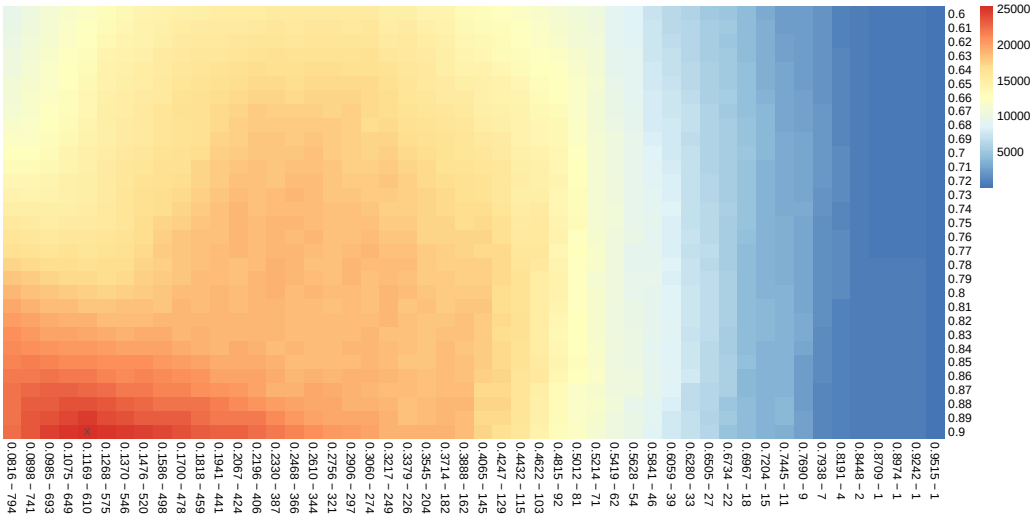
```
# Running stability selection
t0=Sys.time()
out=CalibrateNetwork(data=proteins, K=100, tau=0.5, refine_calib_grid=FALSE, verbose=FALSE)
```

```
t1=Sys.time()
print(t1-t0)
```

```
## Time difference of 20.963 secs
```

As above, the penalty parameter λ and threshold in selection proportion π can be jointly calibrated to maximise the stability score:

```
CalibrationPlot(out)
```

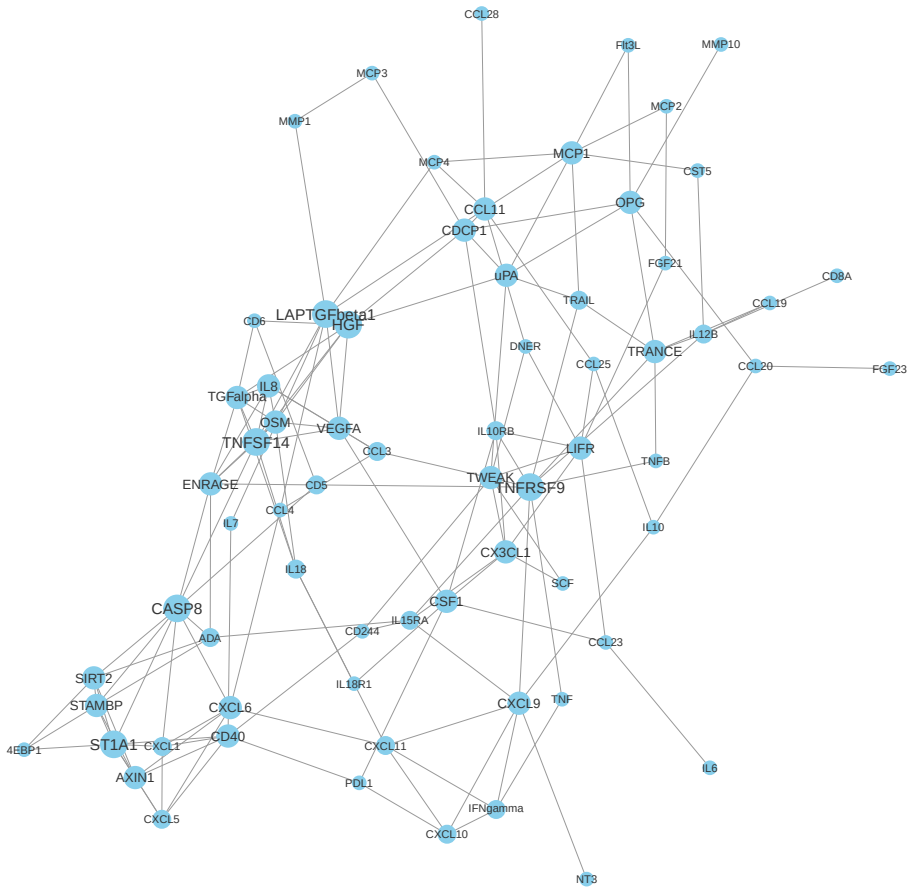


The calibrated network can be obtained as follows:

```
# Adjacency matrix of the calibrated network
A=CalibratedAdjacency(out)

# Getting igraph object
mygraph=GetGraph(adjacency=A)

par(mar=rep(0,4))
set.seed(1)
plot(mygraph, layout=layout_with_fr(mygraph))
```



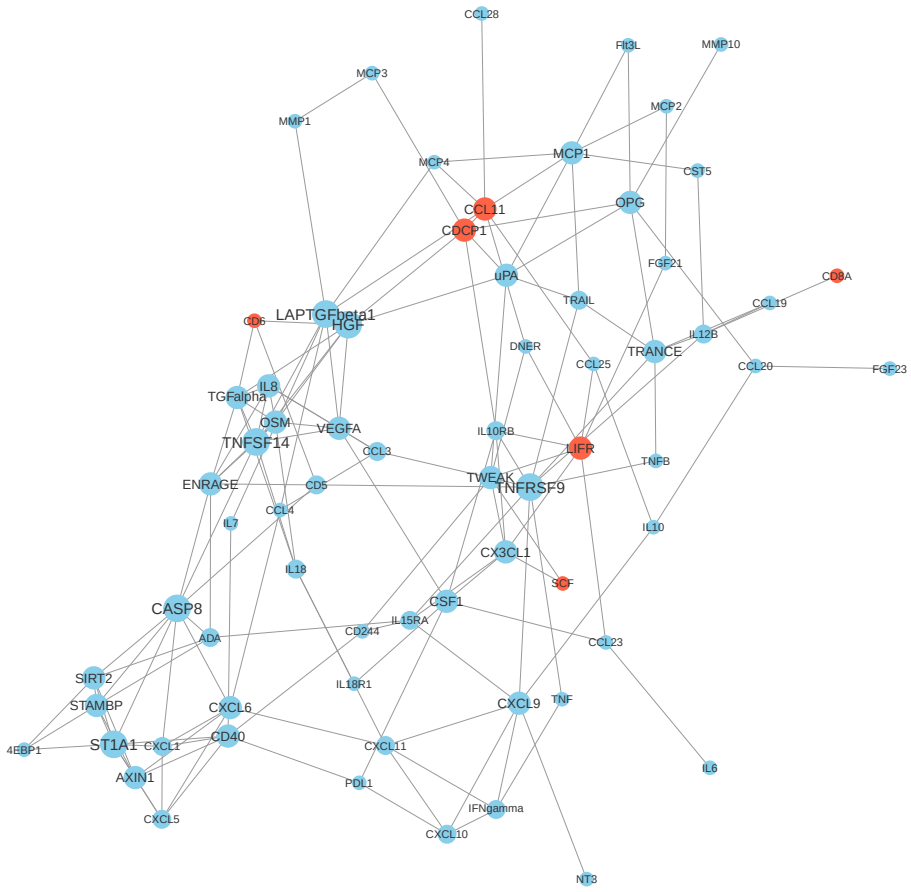
Proteins like CASP8, LAPTGBbeta1 and TNFRSF9 appear quite central in the network. There is no clear modular structure.

The smoking-related proteins identified with the stability selection LASSO in the previous section can be highlighted in the network:

```
mynode_colour=rep("skyblue",ncol(proteins))
names(mynode_colour)=colnames(proteins)
smoking_related_proteins=names(selprop)[selprop>=hat_params[2]]
mynode_colour[smoking_related_proteins]="tomato"

mygraph=GetGraph(adjacency=A, node_color=mynode_colour)

par(mar=rep(0,4))
set.seed(1)
plot(mygraph, layout=layout_with_fr(mygraph))
```



The smoking-related proteins (in red) are all located in the top part of the network. They are not always directly connected but are never far away from each other. The distance between the smoking-related proteins can be computed as follows:

```
print(distances(mygraph)[smoking_related_proteins,smoking_related_proteins])
```

```
##        CDCP1 CCL11 LIFR CD8A CD6 SCF
## CDCP1      0     1    2    4   2   3
## CCL11      1     0    2    4   3   3
## LIFR       2     2    0    4   4   2
## CD8A       4     4    4    0   4   5
## CD6        2     3    4    4   0   4
## SCF        3     3    2    5   4   0
```

All smoking-related proteins except CD8A are one or two edges away from each other.

CONSTRAINED CALIBRATION (INTRODUCING A THRESHOLD IN PFER)

Network models can quickly become very complex in terms of the number of pairs to consider. With p nodes, a total of $\frac{p\times(p-1)}{2}$ pairs of nodes (potential edges) need to be investigated. To prioritise the most reliable edges, and further limit the number of False Positives in the estimated network, the calibration can

be done under the constraint that the upper-bound of the Per-Family Error Rate (PFER, that is the expected number of False Positives) is below a user-defined threshold. In practice, this means that the set of pairs of parameters (λ, π) will be limited to those that yield models with an upper-bound in PFER below the user-defined threshold. This can be done using the `PFER_thr` argument in the function:

```
# Running stability selection under constraint
t0=Sys.time()
out=CalibrateNetwork(data=proteins, K=100, tau=0.5, PFER_thr=20,
                     refine_calib_grid=FALSE, verbose=FALSE)
```

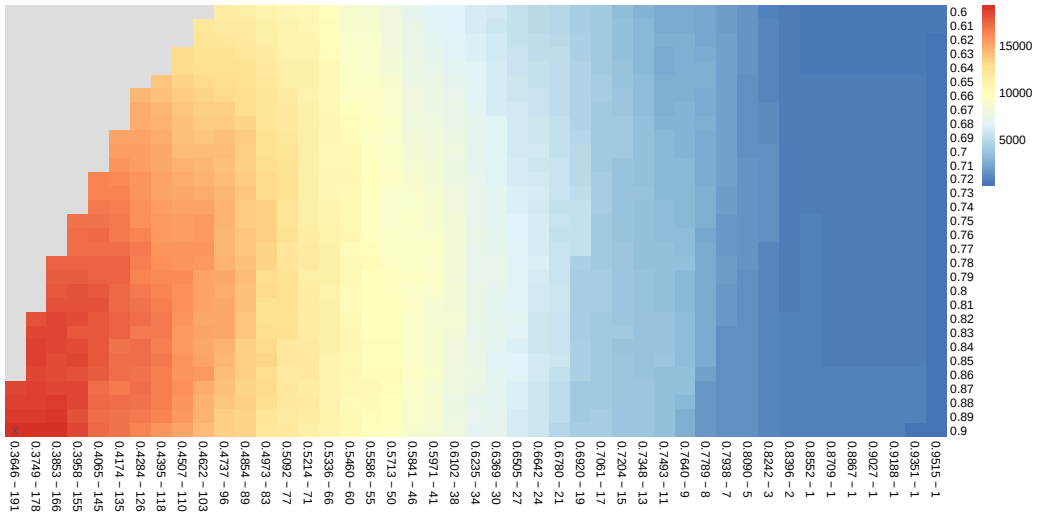
```
## [1] 20
```

```
t1=Sys.time()
print(t1-t0)
```

```
## Time difference of 15.31445 secs
```

In this model, the constraint ensures that the expected number of False Positives is below 20. In the calibration plot, the forbidden area (pairs of parameters generating models with potentially more than 20 FP) is represented in grey:

```
CalibrationPlot(out)
```

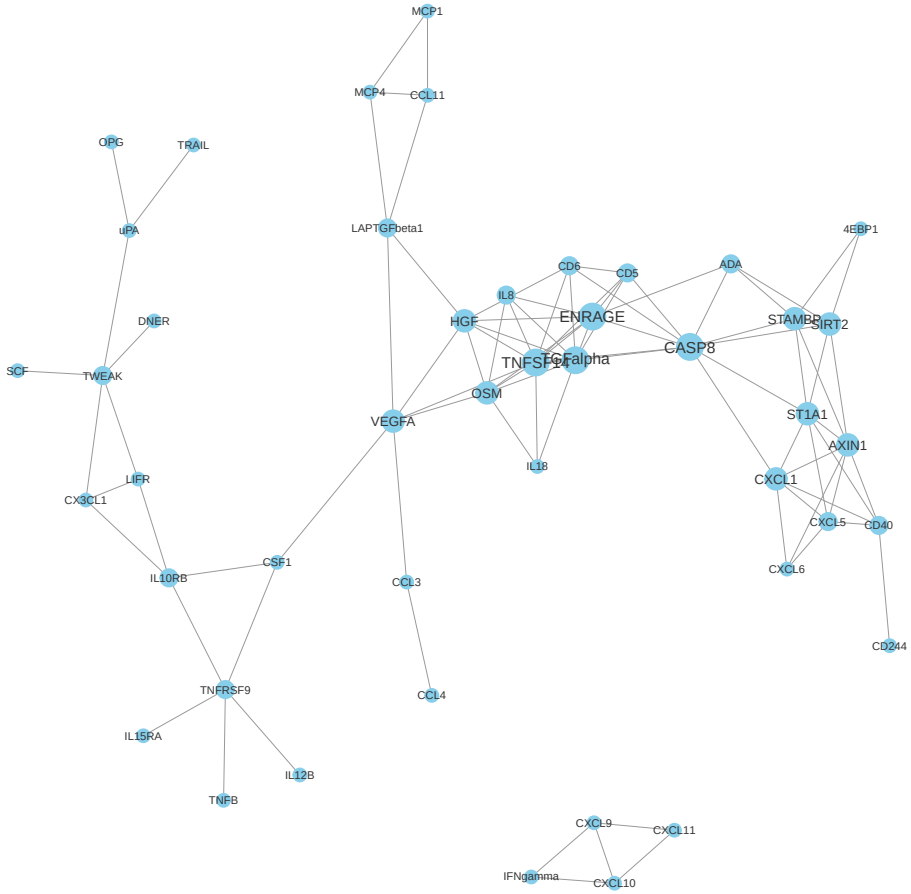


The network calibrated under this constraint can be obtained using the same code as above:

```
# Adjacency matrix of the calibrated network
A=CalibratedAdjacency(out)

# Getting igraph object
mygraph=GetGraph(adjacency=A)
```

```
par(mar=rep(0,4))
set.seed(1)
plot(mygraph, layout=layout_with_fr(mygraph))
```



As expected, the network is sparser when using a constraint on the PFER. The proteins CASP8 and TNFRSF9 are still quite central. In addition, VEGFA seems to play an important role as it is connecting different parts of the network (high betweenness centrality):

```
print(sort(betweenness(mygraph), decreasing=TRUE))
```

```
##       VEGFA        CSF1       CASP8     TNFSF14      IL10RB       TWEAK
##  444.060606  364.000000  304.322872  289.931530  264.000000  187.000000
##     TNFRSF9 LAPTGFbeta1        LIFR       CX3CL1         HGF       CXCL1
##  117.000000  114.000000  102.000000  102.000000   88.556061   86.809885
##         uPA        ST1A1      ENRAGE        CCL3        CD40         OSM
##   79.000000   60.976551   46.981385   40.000000   40.000000   36.966667
##    TGFalpha       SIRT2      STAMBP        MCP4       CCL11         CD6
##   30.004257   29.690115   29.690115   19.500000   19.500000   11.949495
##       AXIN1         ADA       CXCL5       CXCL9      CXCL10         CD5
##   11.166667    6.693795    1.000000    0.500000    0.500000    0.200000
```

```
##          IL8       CD244         OPG        MCP1      CXCL11       TRAIL
##     0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
##         CCL4         SCF        IL18      IL15RA       IL12B       CXCL6
##     0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
##        4EBP1        DNER    IFNgamma        TNFB
##     0.000000    0.000000    0.000000    0.000000
```

MULTI-OMICS NETWORK

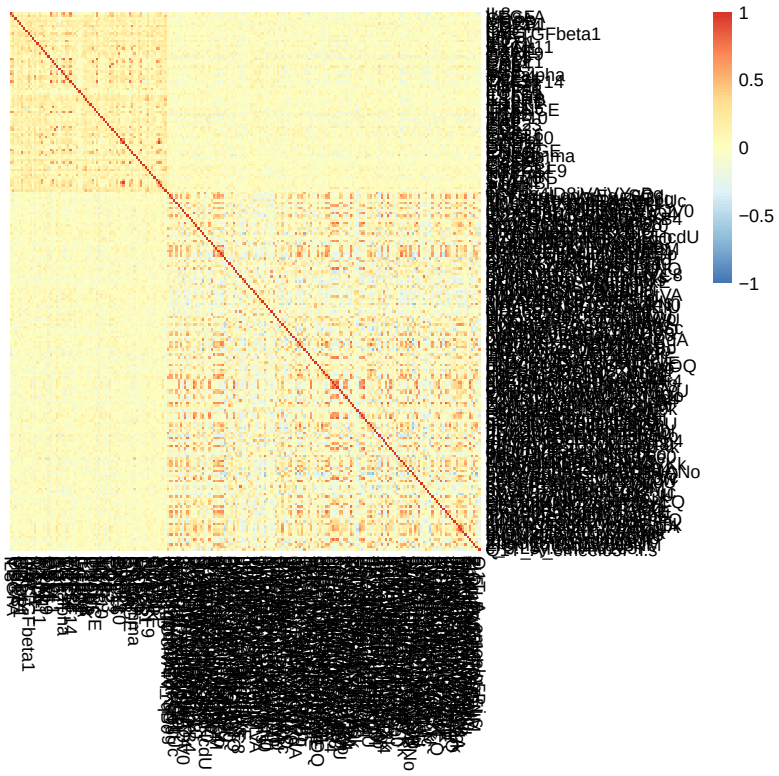
BLOCK STRUCTURE IN THE CORRELATIONS

The inflammatory proteins and smoking-related transcripts can be integrated in a multi-OMICs network. When combining measurements from different OMICs platforms, the correlation matrix often shows a block structure, where within-platform correlations are generally higher than between-platform correlations:

```
# Loading the data
proteins=readRDS("Data/Proteins_selected_denoised_re.rds")
transcripts=readRDS("Data/Transcripts_log_transformed.rds")
annot=readRDS("Data/Transcripts_annotation.rds")

# Combining measurements from the same individuals
ids=intersect(rownames(proteins), rownames(transcripts))
proteins=proteins[ids,]
transcripts=transcripts[ids,]
x=cbind(proteins, transcripts)

# Heatmap of correlation
pheatmap(cor(x), cluster_rows=FALSE, cluster_cols=FALSE, breaks=seq(-1,1,length.out=100), border=NA)
```



BLOCK CALIBRATION

To account for this heterogeneity in the data, we can use block-specific pairs of parameters (λ, π). In this example, we can use three pairs of parameters: one pair for each of the protein-protein, transcript-transcript, and protein-transcript edges.

This can be done using the `pk` argument in the `CalibrateRegression()` function. This argument needs to be a vector the number of variables in each of the platforms. In this example, `pk` is set as a vector of length 2 with the numbers of proteins and transcripts (the order is defined by the way they are merged in the dataset `x`).

The code below can be used to run the multi-block calibration. In this case, the grid of λ values is defined separately using the `LambdaGridNetwork()` function. To speed-up computations, we use a smaller grid with 20 λ values (50 by default in previous sections), as defined in the argument `lambda_path_refined_cardinal`:

```
# Running stability selection under constraint
t0=Sys.time()
pk=c(ncol(proteins), ncol(transcripts))
Lambda=LambdaGridNetwork(data=x, pk=pk, lambda_path_refined_cardinal=20, PFER_thr=50)
```

```
## [1] "Threshold(s) in PFER:"
##  1  2  3
```

```
##  6 23 23
## [1] 50
```

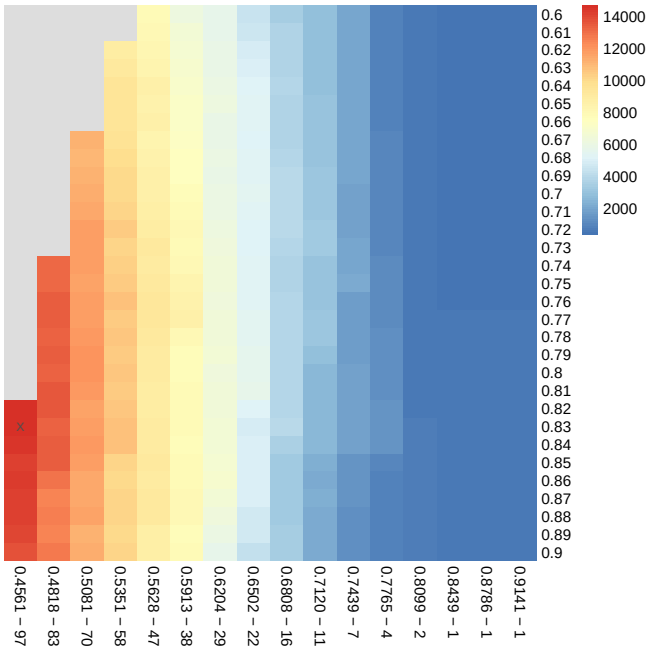
```
out=CalibrateNetwork(data=x, pk=pk, Lambda=Lambda, lambda_other_blocks=apply(Lambda,2,min),
                     K=100, tau=0.5, PFER_thr=50,
                     refine_calib_grid=FALSE, verbose=FALSE)
t1=Sys.time()
print(t1-t0)
```

```
## Time difference of 2.384785 mins
```

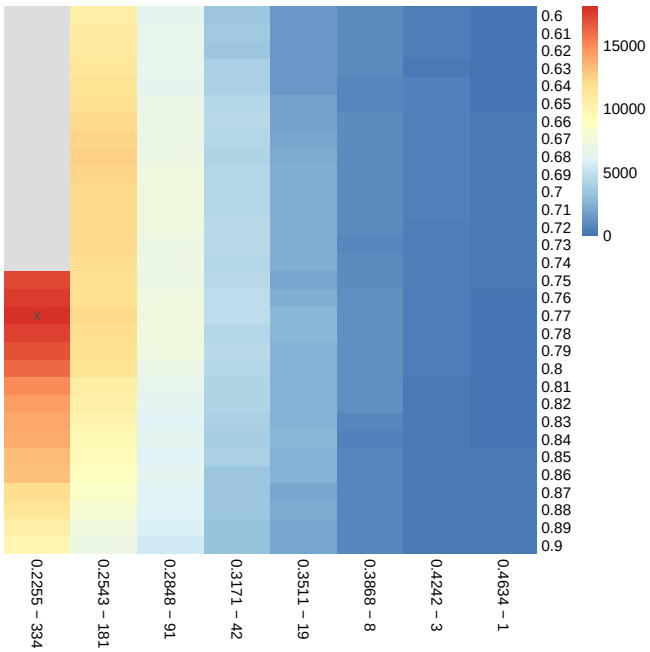
The three calibration plots (one per block) can be visualised using the code below:

```
block_names=c("Protein-protein", "Protein-transcript", "Transcript-transcript")
for (i in 1:3){
  print(block_names[i])
  CalibrationPlot(out, bi_dim = TRUE, block_id=i)
}
```

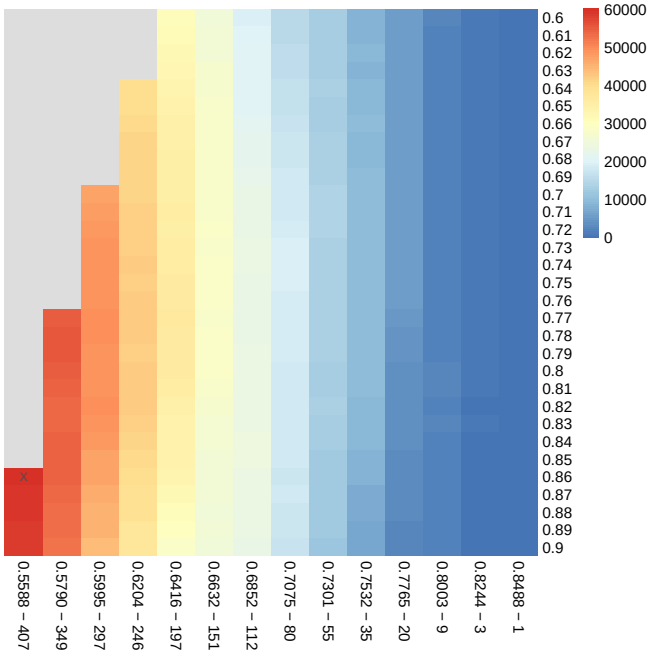
```
## [1] "Protein-protein"
```




```
## [1] "Protein-transcript"
```




```
## [1] "Transcript-transcript"
```



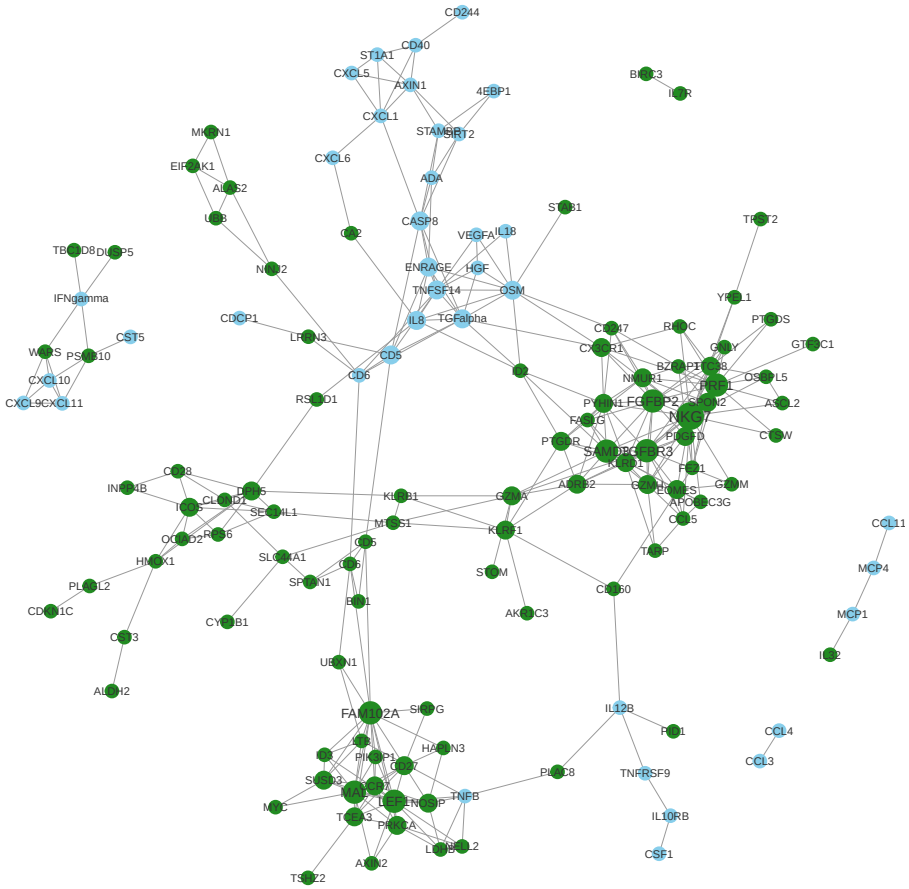
The calibrated multi-OMICs network can be visualised as follows:

```
# Adjacency matrix of the calibrated network
A=CalibratedAdjacency(out)

# Using different colours for the proteins and transcripts
mynode_colours=c(rep("skyblue",ncol(proteins)),rep("forestgreen",ncol(transcripts)))
```

```
# Using gene symbols as labels (transcripts)
mynode_labels=c(colnames(proteins), annot[colnames(transcripts), "Symbol"])
names(mynode_labels)=colnames(x)

# Getting igraph object
mygraph=GetGraph(adjacency=A, node_color=mynode_colours, node_label=mynode_labels)
par(mar=rep(0,4))
set.seed(1)
plot(mygraph, layout=layout_with_fr(mygraph))
```



The network includes many within-platform edges, but also some cross-OMICs edges. In particular, the protein INFB appears quite central in the module of transcripts at the bottom. The connected component on the left is made of proteins (CXCL9, CXCL10, CXCL11 and IFNgamma) and smoking-related transcripts. At the top of the network, CD5 and CD6 are connected to a module of inflammatory proteins and a module of smoking-related transcripts.

The numbers of edges within each of the blocks can be computed as follows:

```
block_mat=GetBlockMatrix(pk=pk)
for (i in 1:3){
  print(block_names[i])
  print(sum(A[block_mat==i])/2)
}
```

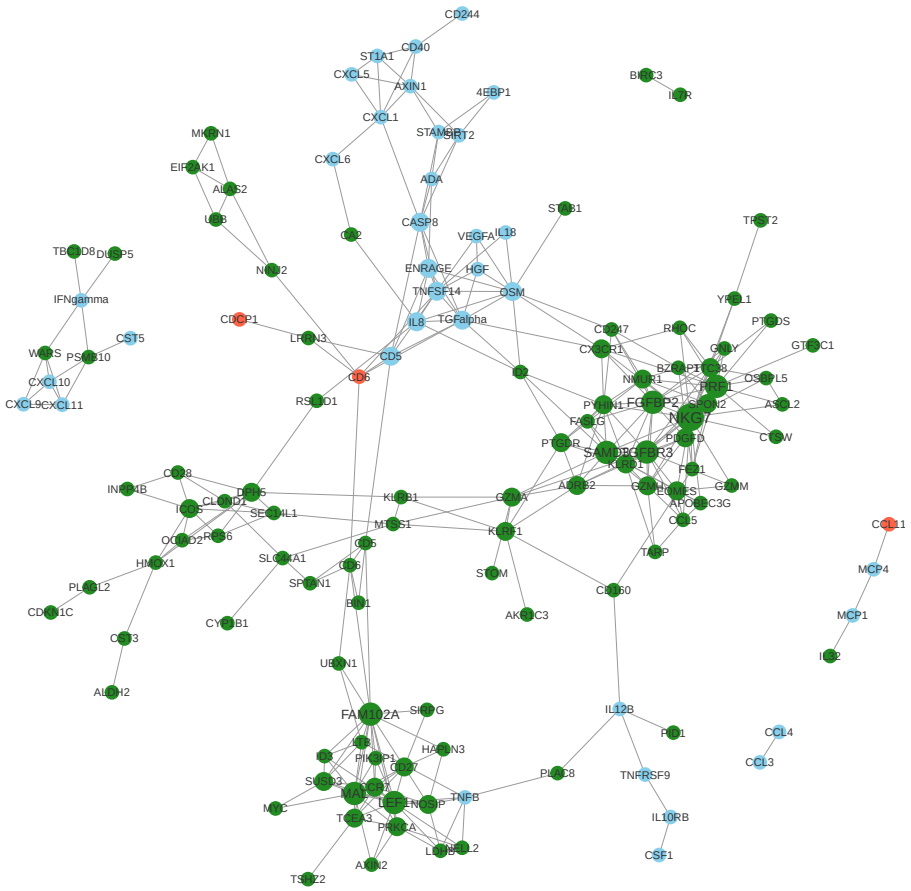
```
## [1] "Protein-protein"
## [1] 60
## [1] "Protein-transcript"
## [1] 36
## [1] "Transcript-transcript"
## [1] 237
```

Challenge 3: Using code from previous sections, colour the smoking-related proteins in red in the network. What can you say about their relationships?

The colour of the smoking-related proteins can be changed using the code below:

```
# Setting node colours
mynode_colours=c(rep("skyblue",ncol(proteins)),rep("forestgreen",ncol(transcripts)))
names(mynode_colours)=colnames(x)
mynode_colours[smoking_related_proteins]="tomato"

# Getting igraph object
mygraph=GetGraph(adjacency=A, node_color=mynode_colours, node_label=mynode_labels)
par(mar=rep(0,4))
set.seed(1)
plot(mygraph, layout=layout_with_fr(mygraph))
```



Three of the six smoking-related proteins (CDCP1, CD6 and CCL11) are included in the network (i.e. they have at least one edge). Two of them are directly connected to the transcript with strongest association with smoking status (LRRN3). The protein CCL11 is connected to a smoking-related transcript through MCP1 and MCP4.